Kódování záporných čísel ve dvojkové soustavě

Přímý kód

- první bit zleva je vyhrazen pro zápis znaménka
- 1 značí záporné číslo
- 0 značí kladné číslo
- změna rozsahu jednoho byte z hodnot 0 - 255 kombinací na -127 až 127
- existují dvě reprezentace nuly +0 a -0
- snadno se dá stanovit absolutní hodnota čísla
- např.: $123_{(10)}$ $01111011_{(2)}$
  $-123_{(10)}$ $11111011_{(2)}$
  $0_{(10)}$ $00000000_{(2)}$
  $-0_{(10)}$ $11111111_{(2)}$

Inverzní kód

- kladná čísla jsou reprezentována normálním způsobem
- je-li číslo záporné provede se na klasickou reprezentaci čísla bitová negace
- existují dvě reprezentace nuly +0 a -0
- např.: $123_{(10)}$ $01111011_{(2)}$
  $-123_{(10)}$ $10000100_{(2)}$
  $0_{(10)}$ $00000000_{(2)}$
  $-0_{(10)}$ $11111111_{(2)}$

Doplňkový kód

- je-li číslo záporné provede se na klasickou reprezentaci čísla bitová negace a číslo zvětšíme o 1
- jediná interpretace nuly
- např.: $123_{(10)}$ $01111011_{(2)}$
  $-123_{(10)}$ $10000101_{(2)}$

Aditivní kód

- nazýváme ho také kód s posunutou nulou
- obraz nuly není nula ve dvojkové soustavě, ale číslo, kde je na začátku nula a dále samé jedničky
- obraz čísla v kódu s posunutou nulou se dostane tak, že k binárnímu vyjádření čísla přičteme obraz nuly
- kladná čísla tedy mají na začátku 1
- záporná čísla a nula mají na začátku 0
- obraz záporných čísel dostaneme provedením bitové negace binárního čísla a za první číslici nulu dosadíme 0
- pro příklad budeme pracovat s 8 bity čísla
- např.: $53_{(10)}$ $110101_{(2)}$
  1. nejprve hodnotu rozšíříme na osm bitů 00110101
  2. nyní vezmeme obraz nuly tedy 01111111
  3. sečteme 00110101 + 01111111
  4. a výsledek je 10110100 tedy číslo 53 je vyjádřeno pomocí doplňkového kódu
  5. záporné číslo -53 bude mít stejnou kroky jako kladné číslo a navíc změníme první

jedničku v obrazu na 0 tedy z 10110100 na 00110100

- používá se pro interpretaci s zápornými čísly v paměti počítače, protože nejsou nutné obvody pro testování čísla lze s číslem normálně pracovat na rozdíl od předešlých způsobů

Zobrazení čísel v pohyblivé řádové čárce

- pro zobrazení velkých nebo desetinných čísel
- používá v moderních počítačích
- jako součástí standardu IEEE 754 na 4 bytech
- čísla jsou zobrazována ve tvaru: $c=M*z^E$
  - M - mantisa čísla, která je zobrazená v soustavě o základu z
  - E – exponent
  - z – základ, který se používá pro výpočet exponentové části

Zobrazení čísla v jednoduché přesnosti

Mantisa

- Je uložena na 23 bitech v přímém kódu se znaménkem.
- Znaménkový bit mantisy je označen *z*.
- Kladné číslo má znaménkový bit nulový, u záporného čísla je v *z* uložena 1.
- Nejvyšší bit mantisy je vždy 1 a nezobrazuje se (mantisa se ukládá počínaje druhým významným bitem – ještě zvyšuje přesnost zobrazení).
- Myšlená desetinná tečka je umístěna za nejvyšším bitem mantisy.

Exponent

- Je uložen na 8 bitech v kódu s posunutou nulou, báze posunutí je $2^7 - 1 = 127$.
- Toto zobrazení čísel v plovoucí řádové čárce používá Java v datovém typu float.
- Např: $53_{(10)}$
  1. převedeme do dvojkové soustavy $110101_{(2)}$
  2. převedeme na normalizovaný tvar a to tak, že před desetinou čárkou bude pouze 1 a podle toho kolik jsme posunuli míst za desetinou čárku tak tolikrát základ soustavy umocníme. V našem případě používáme dvojkovou soustavu proto 2^5
     $1{,}10101 * 2^5$
  3. Exponent spočítáme tak, že vezmeme bázi posunutí a přičteme počet posunutých hodnot za desetinou čárkou:
     2^7-1+5 = 10000100
  4. 53 je kladné číslo tzn znaménkový bit bude 0
  5. nyní již jen zkombinujeme znaménkový bit, exponent a mantisu (hodnoty z normalizovaného tvaru za desetinou tečkou)
     01000010 010101
  6. Doplníme nulami na 32 bitů.
     01000010 01010100 00000000 00000000

Převod desetinného čísla do dvojkové soustavy

Desetinné číslo či jeho část převedeme tak, že ho neustále násobíme dvěma a odečítáme a zapisujeme jedničku, pokud je výsledek větší nebo roven jedné, jinak zapisujeme nulu. Převod končí, pokud nám zbude nula nebo pokud se nám vygeneruje více číslic, než jsme schopni zobrazit.

$0{,}1_{(10)}$

$0{,}1 \cdot 2 = 0{,}2$; 0
$0{,}2 \cdot 2 = 0{,}4$; 0
$0{,}4 \cdot 2 = 0{,}8$; 0

0,8 · 2 = 1,6 ; 1
0,6 · 2 = 1,2 ; 1
0,2 · 2 = 0,4 ; 0
0,4 · 2 = 0,8 ; 0
0,8 · 2 = 1,6 ; 1
... takto by šlo postupovat do nekonečna
nyní již jen číslo přepíšeme do řádku 0,00011001...)$_{(2)}$

ASCII

- z anglického American Standard Code for Information Interchange (americká norma – kód pro výměnu informací)
- jedná se o kódovou tabulku
- v tabulce jsou zapsány:
  - malá/velká písmena anglické abecedy
  - číslice
  - závorky
  - matematické znaky (+ - * / % atd.)
  - interpunkční znaménka (, . : ; atd.)
  - speciální znaky (@ $ ~ atd.))
  - řídící (netisknutelné) kódy
- původní ASCII tabulka využívala 7 bitové kódy, obsahovala tedy 128 znaků
- bylo potřeba zakódovat i národní abecedy, proto byl přidán osmý bit vzniklo tedy dalších 128 kódů a celkem jde do ASCII tabulky zakódovat 256 znaků
- jde tedy o kombinace 8 jedniček a nul
- základní rozdělení kódů ASCII tabulky
  - 0 – 32 jsou přiřazeny řídícím kódům a speciálním znakům
  - 33 – 127 jsou přiřazeny anglické abecedě (malá, velká písmena), číslům, závorkám....
  - 128 – 255 jsou přiřazeny pro kódování národních znaků
- význam kódů 128-255 není tedy jednoznačný
- kódové tabulky ASCII:
  - Windows-1250 kód používaný firmou Microsoft v operačních systémech Windows pro kódování středoevropských jazyků
  - ISO 8859-2 standard ISO, používaný např. v operačním systému Linux
  - CP852 (Latin2) - kód firmy IBM používaný např. v operačním systému DOS. Windows CZ ho využívají při zadávání speciálních znaků pomocí alt-kódů. Např. když stisknete pravý alt, vypíšete na numerické klávesnici číslo 246 a klávesu alt pustíte, objeví se na obrazovce znak ÷